

## AMENDMENTS TO THE CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application:

### Listing of Claims:

- 1           1. (Currently amended) A method for selectively monitoring store  
2 instructions to support transactional execution of a process, comprising:  
3           starting a transactional execution of a block of instructions in a program,  
4           wherein starting the transactional execution involves executing an explicit  
5           ~~performing an~~ instruction implemented in hardware to start the transactional  
6           execution;  
7           encountering a store instruction during the transactional execution,  
8           wherein changes made during the transactional execution are not committed to the  
9           architectural state of a processor until the transactional execution successfully  
10          completes;  
11          determining whether the store instruction is a monitored store instruction  
12          or an unmonitored store instruction by analyzing the store instruction;  
13          if the store instruction is a monitored store instruction,  
14                  performing a corresponding store operation, and  
15                  store-marking a cache line associated with the store  
16                  instruction to facilitate subsequent detection of an interfering data  
17                  access to the cache line from another process; and  
18          if the store instruction is an unmonitored store instruction, performing the  
19          corresponding store operation without store-marking the cache line.

1           2. (Original) The method of claim 1, wherein prior to executing the  
2 program, the method further comprises generating the instructions for the  
3 program, wherein generating the instructions involves:  
4           determining whether store operations that take place during transactional  
5 execution need to be monitored;  
6           generating monitored store instructions for store operations that need to be  
7 monitored; and  
8           generating unmonitored store instructions for store operations that do not  
9 need to be monitored.

1           3. (Original) The method of claim 2, wherein determining whether a store  
2 operation needs to be monitored can involve examining a data structure associated  
3 with the store operation to determine whether the data structure is a “protected”  
4 data structure for which stores need to be monitored, or an “unprotected” data  
5 structure for which stores do not need to be monitored.

1           4. (Original) The method of claim 2, wherein determining whether a store  
2 operation needs to be monitored can involve determining whether the store  
3 operation is directed to a heap, wherein stores from the heap need to be monitored  
4 and stores from outside the heap do not need to be monitored.

1           5. (Original) The method of claim 2, wherein determining whether a store  
2 operation needs to be monitored can involve allowing a programmer to determine  
3 if the store operation needs to be monitored.

1           6. (Currently amended) The method of claim 1, wherein determining  
2 whether the store instruction is a monitored store instruction involves examining  
3 an op code of the store instruction.

1 | 7. (Currently amended) The method of claim 1, wherein determining  
2 whether the store instruction is a monitored store instruction involves examining  
3 an address associated with the store instruction to determine whether the address  
4 falls within a range of addresses for which stores are monitored.

1 8. (Original) The method of claim 7, wherein examining the address  
2 involves comparing the address with one or more boundary registers.

1 9. (Original) The method of claim 7, wherein examining the address  
2 involves examining a Translation Lookaside Buffer (TLB) entry associated with  
3 the address.

1 10. (Original) The method of claim 1, wherein if an interfering data access  
2 from another process is encountered during transactional execution of the block of  
3 instructions, the method further comprises:  
4 discarding changes made during the transactional execution; and  
5 attempting to re-execute the block of instructions.

1 11. (Original) The method of claim 1, wherein if transactional execution of  
2 the block of instructions completes without encountering an interfering data  
3 access from another process, the method further comprises:  
4 committing changes made during the transactional execution to the  
5 architectural state of the processor; and  
6 resuming normal non-transactional execution of the program past the  
7 block of instructions.

1 12. (Original) The method of claim 1, wherein an interfering data access  
2 can include:

3           a store by another process to a cache line that has been load-marked by the  
4 process; and  
5           a load or a store by another process to a cache line that has been store-  
6 marked by the process.

1           13. (Original) The method of claim 1, wherein the cache line is store-  
2 marked in the cache level closest to the processor where cache lines are coherent.

1           14. (Original) The method of claim 1, wherein a store-marked cache line  
2 indicates at least one of the following:  
3           loads from other processes to the cache line should be monitored;  
4           stores from other processes to the cache line should be monitored; and  
5           stores to the cache line should be buffered until the transactional execution  
6 completes.

1           15. (Currently amended) An apparatus that selectively monitors store  
2 instructions to support transactional execution of a process, comprising:  
3           a start transactional execution mechanism configured to start a  
4 transactional execution of a block of instructions in a program, wherein starting  
5 the transactional execution involves executing an explicit ~~performing an~~  
6 instruction implemented in hardware to start the transactional execution;  
7           an execution mechanism within a processor;  
8           wherein the execution mechanism is configured to support the  
9 transactional execution, and wherein changes made during the transactional  
10 execution are not committed to the architectural state of a processor until the  
11 transactional execution successfully completes;  
12           wherein upon encountering a store instruction during transactional  
13 execution, the execution mechanism is configured to,

14                   determine whether the store instruction is a monitored store  
15                   instruction or an unmonitored store instruction by analyzing the  
16                   store instruction,  
17                   if the store instruction is a monitored store instruction, to  
18                   perform a corresponding store operation, and to store-mark a cache  
19                   line associated with the store instruction to facilitate subsequent  
20                   detection of an interfering data access to the cache line from  
21                   another process; and  
22                   if the store instruction is an unmonitored store instruction,  
23                   to perform the corresponding store operation without store-  
24                   marking the cache line.

1           16. (Original) The apparatus of claim 15, further comprising an instruction  
2           generation mechanism configured to:  
3           determine whether store operations that take place during transactional  
4           execution need to be monitored;  
5           generate monitored store instructions for store operations that need to be  
6           monitored; and to  
7           generate unmonitored store instructions for store operations that do not  
8           need to be monitored.

1           17. (Original) The apparatus of claim 16, wherein the instruction  
2           generation mechanism is configured to determine whether a store operation needs  
3           to be monitored by examining a data structure associated with the store operation  
4           to determine whether the data structure is a “protected” data structure for which  
5           stores need to be monitored, or an “unprotected” data structure for which stores do  
6           not need to be monitored.

1           18. (Original) The apparatus of claim 16, wherein the instruction  
2     generation mechanism is configured to determine whether a store operation needs  
3     to be monitored by determining whether the store operation is directed to a heap,  
4     wherein stores from the heap need to be monitored and stores from outside the  
5     heap do not need to be monitored.

1           19. (Original) The apparatus of claim 16, wherein the instruction  
2     generation mechanism is configured to determine whether a store operation needs  
3     to be monitored by allowing a programmer to determine if the store operation  
4     needs to be monitored.

1           20. (Original) The apparatus of claim 15, wherein the execution  
2     mechanism is configured to determine whether a store operation needs to be  
3     monitored by examining an op code of the store instruction.

1           21. (Original) The apparatus of claim 15, wherein the execution  
2     mechanism is configured to determine whether a store operation needs to be  
3     monitored by examining an address associated with the store instruction to  
4     determine whether the address falls within a range of addresses for which stores  
5     are monitored.

1           22. (Original) The apparatus of claim 21, wherein the execution  
2     mechanism is configured to examine the address by comparing the address with  
3     one or more boundary registers.

1           23. (Original) The apparatus of claim 21, wherein the execution  
2     mechanism is configured to examine the address by examining a Translation  
3     Lookaside Buffer (TLB) entry associated with the address.

1           24. (Original) The apparatus of claim 15, wherein if an interfering data  
2 access from another process is encountered during transactional execution of the  
3 block of instructions, the execution mechanism is configured to:  
4           discard changes made during the transactional execution; and to  
5           attempt to re-execute the block of instructions.

1           25. (Original) The apparatus of claim 15, wherein if transactional  
2 execution of the block of instructions completes without encountering an  
3 interfering data access from another process, the execution mechanism is  
4 configured to:  
5           commit changes made during the transactional execution to the  
6 architectural state of the processor; and to  
7           resume normal non-transactional execution of the program past the block  
8 of instructions.

1           26. (Original) The apparatus of claim 15, wherein an interfering data  
2 access can include:  
3           a store by another process to a cache line that has been load-marked by the  
4 process; and  
5           a load or a store by another process to a cache line that has been store-  
6 marked by the process.

1           27. (Original) The apparatus of claim 15, wherein the cache line is store-  
2 marked in the cache level closest to the processor where cache lines are coherent.

1           28. (Original) The apparatus of claim 15, wherein a store-marked cache  
2 line indicates at least one of the following:  
3           loads from other processes to the cache line should be monitored;

4 stores from other processes to the cache line should be monitored; and  
5 stores to the cache line should be buffered until the transactional execution  
6 completes.

1 29. (Currently amended) A computer system that selectively monitors  
2 store instructions to support transactional execution of a process, comprising:  
3 a processor;  
4 a memory;  
5 a start transactional execution mechanism within the processor configured  
6 to start a transactional execution of a block of instructions in a program, wherein  
7 starting the transactional execution involves executing an explicit ~~performing an~~  
8 instruction implemented in hardware to start the transactional execution;  
9 an execution mechanism within the processor;  
10 wherein the execution mechanism is configured to support the  
11 transactional execution, wherein changes made during the transactional execution  
12 are not committed to the architectural state of a processor until the transactional  
13 execution successfully completes;  
14 wherein upon encountering a store instruction during transactional  
15 execution, the execution mechanism is configured to,  
16 determine whether the store instruction is a monitored store  
17 instruction or an unmonitored store instruction by analyzing the  
18 store instruction,  
19 if the store instruction is a monitored store instruction, to  
20 perform a corresponding store operation, and to store-mark a cache  
21 line associated with the store instruction to facilitate subsequent  
22 detection of an interfering data access to the cache line from  
23 another process; and



24           if the store instruction is an unmonitored store instruction, to perform the  
25   corresponding store operation without store-marking the cache line.